AMENDMENTS TO THE SPECIFICATION:

Please amend the paragraph beginning on page 19, line 15 as follows:

Figure 3 illustrates an embodiment of the present invention used to set the score threshold 104 in step 110. In step 301, documents from the training set or subset of the training dataset, possibly not overlapping with the subset of the dataset used for term extraction, referred to as the thresholding dataset, are scored against the profile vector, and are sorted in descending order according to their scores. At each position in the ranked list, a utility value U_i can be computed by assuming a threshold that is equal to the score of the document at that position. Therefore, in step 302, each position yields a candidate score threshold and a corresponding utility value. Thereafter, the "optimal" utility threshold, θ_{opt} 291 (491) is determined in step 303 as the score where the utility is maximum over the thresholding dataset and the "zero" utility threshold, θ_{zero},292 (492) is determined in step 304 to be the highest score below θ_{opt} 291 (491) where the utility is zero or negative (or the lowest score should the utility fail to reach zero). Using the optimal utility threshold and the zero utility threshold, a new profile utility threshold is then calculated in step 305 by interpolating between the empirical optimal utility threshold and the zero utility threshold over the thresholding dataset using a beta-gamma function 306 as follows:

$$threshold = \alpha * \theta_{zero} + (1-\alpha) * \theta_{opt}$$
$$\alpha = \beta + (1-\beta) * e^{-M*\gamma}$$

Please amend the paragraph beginning on page 25, line 9 as follows:

A *multiplex filter* 591 with three element or constituent filters F_i 520, 525, and 526, is illustrated in Figure 5a. (A multiplex filter is not limited to three constituent filters as given for illustration in Figure 5a, rather can consist of *i* such filters, for any *i*.) This multiplex filter 591, made up of constituent filters 520, 525, and 526, accepts or rejects a document 510 (where document 510 is represented in terms of its features as defined above) based on some interpretation of the independent scoring of each constituent filter F_i. That is, each component filter 520, 525, and 526 accepts as input the features and associated values that describe the document 510 and scores them against the component filter profiles. The individual filter scores 570, 575, 580 are then aggregated <u>to produce an output 585</u> using a function 595. Various aggregation functions 595 can be used for interpreting the scores of a set of filters 570, 575 and 580, ranging from some simple combination of binary outcomes (e.g., the sum of the "votes" of each filter) to a weighted, possibly non-independent scoring based on the interaction of filters. In general, classification of a document 510, *Doc,* using multiplex filters is based upon the following procedure where each component filter is assigned a weight Wgt_i (e.g., uniform weight or weight proportional to its performance expectation):

$$Class(Doc) = Sign\left(\sum_{t=1}^{T} Wgt_t Model_i(Doc)\right)$$

Please amend the paragraph beginning on page 28, line 17 as follows:

- *Source features* are the features directly provided to the system by the pre-processing of the document. These correspond to columns 530c. Therefore, column 510c, for example, corresponds to the source feature scores $F1_D$ for each document.

Please amend the paragraph beginning on page 28, line 19 as follows:

- *Derived features* that are computed by filters earlier in the stack of filters. In our example, these earlier or lower level filters (models) are 510b, 515b and 520b. Each of these features is computed by the filter from which it is derived. That is each lower level filter (e.g., 510b) processes (scores) each example 535b in the training database. In Figure 5c, row 515c, for example, represents document DOC ID 1. This can result in a binary value (shown as a positive output 540b and a negative 545b output) or an actual score (that is, in this case, the document 505b is scored against the filter and the similarity score taken as the actual score) or both. In the example Figure 5c, for explanation purposes, this is limited to the score value. This process results in adding a column 520c (corresponding to the result of scoring each document against filter 510b) to the training dataset where each cell value corresponds to the score between each document and the model 515b. Therefore, the columns 540c represent the results of scoring the documents against each model M_1 to M_m.

Please amend the paragraph beginning on page 32, line 22 as follows:

A variant of a cascade filter is depicted in FIG. 6. Here, each component filter 630, 635 and 640 accepts as input the source features that describe the document 615, along with derived features from the output of earlier filters in the ensemble 670, 671 and 672. Note that elements 615, 645, 646, 650, 655, 660, 665 and 692 of FIG. 6 have functions corresponding to the functions of elements 515, 545, 546, 547, 555, 560, 565 and 592 of FIG. 5d respectively. Here the output of the previous filter could be the actual score of the document against the filter or a classification value (+1/-1) or both. Note that the information added by the processing score or other assessment by a filter ordered earlier in a sequence can be regarded as a new feature in the feature discrimination space of a subsequent filter. Such new, possibly abstract, features (such as the features 540 illustrated in FIG. 5c) can be exploited by subsequent filters in their training and in their processing of documents generally.

Please amend the paragraph beginning on page 32, line 7 as follows:

The focus of the construction algorithm for cascade filters is on producing a series of filters. The training set used for each filter in the series is chosen based on the performance of earlier filters in the series. A preferred embodiment for constructing a cascade filter for an information need, *T,* involves a number of steps and assumes as input two subsets of the training dataset, *D1* 704, *D2* 702, which are respectively used for feature extraction and threshold optimization. The main steps of the algorithm are outlined in block format in Figures 7 to 11. The algorithm consists of two threads 700 and 701: the extraction thread 700 and the threshold-setting or threshold-optimization thread 701. Each thread results in the construction of its own cascade filter, namely, $C_{Extraction}$ 738 and C_{Opt} 739. The algorithm is iterative in nature, whereby the first filter in the cascade, $C1_{Extraction}$ ~~810~~ 710,is constructed using the positive topic examples in the extraction dataset *D1* ~~804~~ 704. This cascade corresponds to the extraction cascade $C_{Extraction}$ 838. In order to set the threshold for $C1_{Extraction}$, a second cascade filter (i.e., the optimization cascade) 839 is constructed. The first constituent filter ~~820~~ 720 in this cascade is a copy of $C1_{Extraction}$ ~~810~~ 710 and is denoted as $C1_{Opt}$ ~~820~~ 720. To avoid clutter in Figure 8, the *Extraction* and *Opt* suffixes are dropped from the component filters names. The threshold 820 for $C1_{Opt}$ ~~820~~ 720 can be set using any of a number of threshold-setting techniques with respect to a specified utility measure over the *D2* dataset ~~802~~ 702. One such method is the beta-gamma thresholding algorithm described earlier. The threshold 822 of the $C1_{Extraction}$ filter ~~810~~ 710 is set to the optimized threshold 820 of $C1_{Opt}$ ~~820~~ 720. Subsequently, the fallout, or remainder documents from filter *C1* ~~810~~ 710, which pass through the negative class channel 821 (i.e., positive examples from *D1* that are rejected by $C1_{Extraction}$) are used to construct the second filter $C2_{Extraction}$ ~~930~~ 711 in the cascade, provided various continuation conditions are met. These continuation conditions may include one or more of (but not limited to) the following: the number of documents in the fallout, or remainder of $C1_{Extraction}$ ~~822 (not shown)~~, graphically depicted in Figure 9 as 922, is greater than a minimum number of documents required to construct a filter; the utility of the $C1_{Opt}$ ~~821~~ 720 graphically depicted in Figure 9 as 921 over the optimization dataset is greater than some threshold (e.g., zero). Next, the threshold 932 for $C2_{Opt}$ 721 can be set using a threshold-setting techniques as described above with respect to a specified utility measure over the *D2* dataset 702. The threshold of the $C2_{Extraction}$ filter 711 is set to the optimized threshold 932 of $C2_{Opt}$ 721. Subsequently, the fallout, or remainder documents from filter *C2* 711, which pass through the negative class channel 933 (i.e., positive examples from *D1* that are rejected by $C2_{Extraction}$) are used to construct subsequent filters $Cn_{Extraction}$ 712 with scoring thresholds Cn_{Opt} 722 in the cascade, provided the continuation conditions are met. The above steps of constituent filter extraction and threshold optimization (on the fallout, or remainder of each preceding filter) are repeated as long as the continuation conditions are satisfied, yielding the component filters C1, C2, . . . Cn in the cascade as illustrated in Figure 10. Referring to Figure 11, once any one of the continuation conditions fails, all the positive outputs of the constituent filters of the extraction cascade $C_{Extraction}$ 1115 are connected to a union filter 1152. There are two outputs of the cascade filter 1100 (corresponding to both the positive and negative results of the component filters); the output of the union filter corresponds only to the positive or accepted documents 1151; the fallout, or remainder through the final component filter corresponds to the negative or rejected documents 1150.